AMTH 491
Measuring Entropy in Graph Datasets Using Geometric Scattering and Diffusion Geometry

Selma Mazioud

Advisors: Smita Krishnaswamy and Ian Adelstein

December 2024

Abstract

In this study, we propose a novel method to compute entropy over a set of graphs, providing a statistic that quantifies the variability of a graph dataset. Our approach is based on Diffusion Spectral Entropy (DSE), designed for the analysis of point clouds. Graph Neural Networks (GNNs) have traditionally been considered as a method to embed graphs into point clouds. However, we demonstrate the limitations of this approach and instead propose the use of geometric scattering, a transform that applies wavelet convolutions and non-linear activations to extract hierarchical, invariant features from data. Previous studies have shown that geometric scattering, an untrained embedding method, outperforms vanilla GNNs and produces embeddings that more effectively preserve the geometric and topological structure of the data. We demonstrate that our method effectively captures the variance of a graph dataset through experiments on toy datasets generated by increasingly perturbing a graph. Additionally, we apply our method to a real molecular dataset. The results highlight the potential of using diffusion spectral entropy combined with geometric scattering as a powerful tool for analyzing graph variability. We show our implementation in https://github.com/selma-m/graph-entropy

Keywords: Graph Embeddings, Geometric Scattering, Diffusion Geometry, Graph Neural Networks, Diffusion Spectral Entropy.

Acknowledgments

I would like to thank my co-advisors, Professor Ian Adelstein and Professor Smita Krishnaswamy, for their guidance and support throughout the design and implementation of this project. Their diverse perspectives and expertise have deeply enriched my understanding of the field, and I have truly enjoyed working with them both—learning from their insights and benefiting from their thoughtful feedback. I also want to express my gratitude to Professor Helen Caines—her mentorship and faith in my abilities gave me the confidence and passion for research. Her encouragement to think critically and independently has been invaluable, and I am deeply grateful for her ongoing support throughout my college journey.

I am also grateful to PhD students Chen Liu, Siddharth Viswanath, and Danqi Liao for their invaluable support on this project, including brainstorming ideas and helping with debugging. A special thanks to Postdoctoral Fellow Fernando Flor for his thoughtful mentorship and guidance. Finally, I would like to thank my parents for their endless love, encouragement, and belief in me, which has been a constant source of strength throughout my life.

I dedicate this thesis to my beloved grandfather, whose wisdom, love, and patience have shaped me into the curious learner I am today.

Papa, Maman: c'est aussi pour vous.

Contents

1 Introduction

Graph structures are ubiquitous in representing data across various domains, from social networks to molecular biology. Unlike regular data structures such as grids or sequences, graphs introduce a layer of complexity due to their irregularity and the rich structural information encoded by their nodes and edges. This complexity poses significant challenges when attempting to compute standard statistics, such as mean and variance, over graph datasets. Without a canonical method for these computations, important analyses—such as ensuring dataset diversity or studying molecular conformations—become difficult, underscoring the need for novel statistical measures tailored to graph data.

To address this gap, we propose a method to compute entropy over graphs, providing an insightful statistic that quantifies the variability within graph datasets. Specifically, we leverage Diffusion Spectral Entropy (DSE) [8], a method originally designed for the analysis of point clouds. However, applying DSE to graphs necessitates an efficient embedding method to transform graphs into point clouds. A natural initial approach involves using Graph Neural Networks (GNNs) [15], which are designed to exploit the relationships between nodes and edges to learn effective node or graph representations. Despite their utility, traditional GNNs suffer from limitations such as oversmoothing, where the learned representations become indistinguishable across nodes, and oversquashing, where distant nodes' information gets compressed too severely. In addition, a foundational principle of most node-level analysis is homophily, the idea that nodes are similar to their neighbors [17]. This means GNNs aim to produce hidden representations of each node that vary slowly among neighboring nodes. However, for signal-level tasks, this local homophily heuristic is not applicable. It becomes crucial to capture both low-frequency and high-frequency information in the input signal, as well as both local and global behaviors.

Given the limitations of GNNs, we turn to a method inspired from graph signal processing [12]: geometric scattering [5], [4], [18], a generalization of the scattering transform to graphs, as an alternative embedding method. Geometric scattering alternates wavelet transforms with non-linear modulus activations in a multi-order, multi-scale fashion, effectively capturing both low-frequency and high-frequency information, as well as local and global behaviors in the input signal. This method has been shown to outperform traditional GNNs in preserving the geometric and topological structure of graph data and providing a solid foundation for signal-level tasks.

One of the key advantages of having a reliable measure of entropy for graphs lies in its ability to determine whether a set of graphs exhibits homophily or heterophily. This is particularly important for ensuring that GNNs are trained on diverse graph datasets, thereby enhancing their robustness and generalizability. Furthermore, in fields like molecular dynamics, measuring entropy over molecular conformations can distinguish between minor thermal fluctuations and significant conformational changes, offering deeper insights into molecular behavior.

In this study, we generate random graphs with varying node counts and perturbations, equipping them with Dirac signals on each node to demonstrate the efficacy of our approach. Our experiments on both synthetic and real graph datasets reveal the robustness and potential of combining diffusion spectral entropy with geometric scattering as powerful tools for analyzing graph variability.

Our contributions are three-fold:

- We propose a method for computing entropy over a set of graphs.
- We demonstrate that geometric scattering is better able to preserve the dataset's structure than other graph embedding methods like GNNs.

- We show how our framework can be applied to molecular datasets.

2 Background

2.1 Diffusion Maps

Principal Component Analysis (PCA) is a powerful dimension reduction technique, but, as a linear method, cannot be effectively be extended to data with non-linear structures. In this section, we will present two non-linear dimension reduction methods: Diffusion maps and Laplacian eigenmaps.

2.1.1 Diffusion operator

Given a point cloud $\mathbf{X}$ containing n points, the diffusion operator P, also known as the random walk operator, is computed by first calculating the pairwise distances between points and then constructing a similarity or adjacency matrix. We define a kernel function $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ and use it to populate the adjacency matrix $\mathbf{W}$, where each element is defined as $\mathbf{W}_{ij} := \kappa(x_i, x_j)$ for all points $x_i, x_j \in \mathbf{X}$. Here, we employ a Gaussian kernel to construct the adjacency matrix. The resulting weighted graph $\mathcal{G}$, with its set of nodes representing observations and adjacency matrix values in $(0, 1]$, forms the basis for defining the diffusion operator. The diffusion operator is given by

$$\mathbf{P} := \mathbf{D}^{-1}\mathbf{W}, \tag{1}$$

where $\mathbf{D}$ is the degree matrix. This operator is row stochastic and therefore describes a Markovian random walk or diffusion on the graph, making it a fundamental tool for uncovering the structure of datasets in manifold learning.

2.1.2 Diffusion maps

Based on the diffusion operator, [3] defines an embedding in k dimensions via the first k non-trivial right eigenvectors of the t-steps diffusion operator $\mathbf{P}^t$ weighted by their eigenvalues. Let $\{\lambda_i\}_{i\in\mathbb{N}}$ be the eigenvalues of $\mathbf{P}$ and let $\{\psi_i\}_{i\in\mathbb{N}}$ be its eigenvectors. The family of diffusion maps $\{\Psi_t\}_{t\in\mathbb{N}}$ is given by:

$$\Psi_t(x) = \begin{bmatrix} \lambda_1^t \psi_1(x) \\ \lambda_2^t \psi_1(x) \\ \cdots \\ \lambda_n^t \psi_n(x) \end{bmatrix} \tag{2}$$

The embedding preserves the ***diffusion distance***

$$DM_{\mathbf{P}}(x_i, x_j) := \|(\delta_{\mathbf{i}}\mathbf{P}^t - \delta_{\mathbf{j}}\mathbf{P}^t)(1/\pi)\|_2, \tag{3}$$

where δ_i is a vector such that $(\delta_i)_j = 1$ if $j = i$ and 0 otherwise, and π is the stationary distribution of $\mathbf{P}$. Intuitively, $DM_{\mathbf{P}}(x_i, x_j)$ considers all the t-steps paths between x_i and x_j. A larger diffusion time can be seen as a low-frequency graph filter, keeping only information from low-frequency transitions such as the stationary distributions. The diffusion matrix converges to a stationary distribution $\pi_i := \mathbf{Q}_{ii}/\sum_i \mathbf{Q}_{ii}$ under mild assumptions, making diffusion with $t > 1$ useful in denoising relationships between observations.

2.2 Spectral Graph Theory

2.2.1 Graph Laplacian

We consider a (weighted) graph $G = (V, E)$ on n nodes, with adjacency matrix $\mathbf{W}$. The **graph Laplacian** is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W},$$

where $D_{ii} = \sum_j w_{ij}$. It can be regarded as a discretization of the standard normal operator ∇^2 and has several nice properties. L is real symmetric and therefore admits an orthonormal basis $\{\phi_i\}_{i=1}^n$.The Laplacian matrix is a positive semi-definite matrix that always has a 0 eigenvalue and $\mathbf{1}$ associated eigenvector since $\mathbf{L}$ has row sum equal to zero by construction: $\mathbf{L1} = \mathbf{0} = 0 \cdot \mathbf{1}$.

Theorem 2.1. *The Laplacian L is a symmetric positive semi-definite matrix,*

Proof. $\mathbf{L} = \mathbf{D} - \mathbf{W} = \mathbf{D}^\top - \mathbf{W}^\top$ since $\mathbf{D} = \mathbf{D}^\top = \mathbf{L}^\top$ as a diagonal matrix and $\mathbf{W} = \mathbf{W}^\top$ for an undirected graph.
For $\mathbf{x} \in \mathbb{R}^n$, letting $\mathbf{e}_i \in \{0, 1\}^n$ be the standard basis vectors for $i \in [n]$:

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{(i,j)\in E} W_{ij}(\mathbf{x}^\top(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top \mathbf{x}) = \sum_{(i,j)\in E} W_{ij}(x_i - x_j)^2 \geq 0 \tag{4}$$

□

We can then order the eigenvalues of L: $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n \leq 2$. Moreover, the number of zero eigenvalues in the Laplacian $\mathbf{L}$ is the number of connected components of the graph. In particular, G is connected if and only if the Fiedler value λ_2 is strictly positive. We have $\lambda_n = 2$ if and only if the graph G is bipartite.

Theorem 2.2. *The graph G is connected if and only if $\lambda_2 > 0$.*

Proof. ($\Leftarrow$) We proceed my contradiction. Suppose that $\lambda_2 > 0$ and that G is not connected with connected components G_1 and G_2. We can write the Laplacian:

$$\mathbf{L} = \begin{bmatrix} L_{G_1} & 0 \\ 0 & L_{G_1} \end{bmatrix}$$

for which $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ are orthogonal eigenvectors with eigenvalue zero. Then, $\lambda_1 = \lambda_2 = 0$, which contradicts our initial assumption.
($\Rightarrow$) Again, we proceed by contradiction. Suppose G is a connected graph and that $\lambda_2 = 0$. By (1), ϕ_2 is constant across adjacent nodes, and therefore across the graph by connectivity. This yields a contradiction as we must have ϕ_2 orthogonal to $\phi_1 = \mathbf{1}$. □

The normalization of the graph Laplacian is the symmetric matrix

$$\mathcal{L} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = I - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}.$$

There is a strong relationship between the spectra of the normalized Laplacian and the diffusion matrix. In fact, Note that we can relate the diffusion operator $\mathcal{P}$ with the normalized symmetric Laplacian $\mathcal{L}$:

$$\mathcal{L} = \mathbf{I} - \mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2}.$$

Letting $\{\psi_i\}_{i=1}^n$ be the eigenvectors and $\{\omega_i\}_{i=1}^n$ the eigenvalues of $\mathcal{L}$, we can write

$$\psi_j = \mathbf{D}^{1/2}\phi_j,$$
$$\lambda_j = 1 - \omega_j,$$
$$1 = \omega_1 \geq \omega_2 \geq \omega_n \geq 1.$$

The graph G is disconnected if and only if $\omega_2 = 1$ and $omega_n = -1$ if and only if the graph is bipartite.

2.2.2 Graph Signal Processing

Because the graph Laplacian is a discretization of the standard second derivative operator ∇^2, its eigenvalues describe harmonics on a graph. Projecting signals onto such harmonics yields a **graph Fourier transform**. Then, we can extend many of the principles of signal processing to graphs.

In fact, we can write $\mathcal{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ where $\mathbf{V}$ is a matrix whose i-th column is $\mathbf{v}_i$ and $\mathbf{\Lambda}$ is a diagonal matrix with $\mathbf{\Lambda}_{ii} = \lambda_i$. Since $\{\psi_i\}_{i=1}^n$ forms an orthonormal basis, it is easy to show that $\mathbf{V}\mathbf{V}^\top = \mathbf{I}$. As shown in [10], we have that

$$\mathbf{x}^\top \mathcal{L}\mathbf{x} = \sum_{(i,j)\in E} (\tilde{x}_i - \tilde{x}_j)^2,$$

where $\tilde{\mathbf{x}} = \mathbf{D}^{-1/2}\mathbf{x}$ is a normalization of $\mathbf{x}$. In other words, the quadratic form of the normalized Laplacian matrix $\mathcal{L}$ describes the smoothness of (normalized) signals. Therefore, we can interpret its eigenvalues $\{\lambda_i\}_{i=1}^n$ as frequencies and its eigenvectors $\{\psi_i\}_{i=1}^n$ as Fourier modes.

2.3 Diffusion Spectral Entropy

The diffusion matrix of an embedding effectively captures its geometry. Looking at hidden layer embeddings of different neural network architectures, we use an information theoretic measure called Diffusion Spectral Entropy (DSE), which was introduced in [8] to compute a fast entropy on data by first creating a data diffusion operator, and then computing the entropy of the eigenvalues. The DSE is defined as the entropy of the eigenvalues of the diffusion operator $\mathbf{P}_X$ computed on a dataset $\mathbf{X}$.

$$S_D(\mathbf{P}_X, t) := -\sum_i \alpha_{i,t}\log(\alpha_{i,t})$$

where $\alpha_{i,t} := \frac{|\lambda_i^t|}{\sum_j |\lambda_j^t|}$, and $\{\lambda_i\}$ are the eigenvalues of the diffusion matrix $\mathbf{P}_X$ and t is the diffusion time. Defined this way, the DSE captures the spread of point cloud data over its dimensions. A high DSE would correspond to an even spread across diffusion dimensions.

The following theorem is proven in [8] (Proposition 4.2):

Theorem 2.3. *As $t \to \infty$, $S_D(\mathbf{P}_X, t) \to \log(k)$ for data X with k well-separated clusters.*

Proof. Suppose the data X has k well-separated clusters. Then, $\mathbf{P}_X$ has eigenvalues $1 = |\lambda_1| = |\lambda_2| = ... = |\lambda_k| > |\lambda_{k+1}| \geq 0$. In other words, 1 is an eigenvalue of $\mathbf{P}_X$ with multiplicity k. All other eigenvalues are strictly less than 1 and non-negative. Then, as $t \to \infty$, $\lambda_i^t \to 0$ for $i \geq k+1$ and $\lambda_i^t \to 1$ for $i \leq k$. So only the first k eigenvalues remain as the others shrink to 0. The resultant DSE is

$$\sum_k \frac{1}{k}\log(k) = \log(k)$$

□

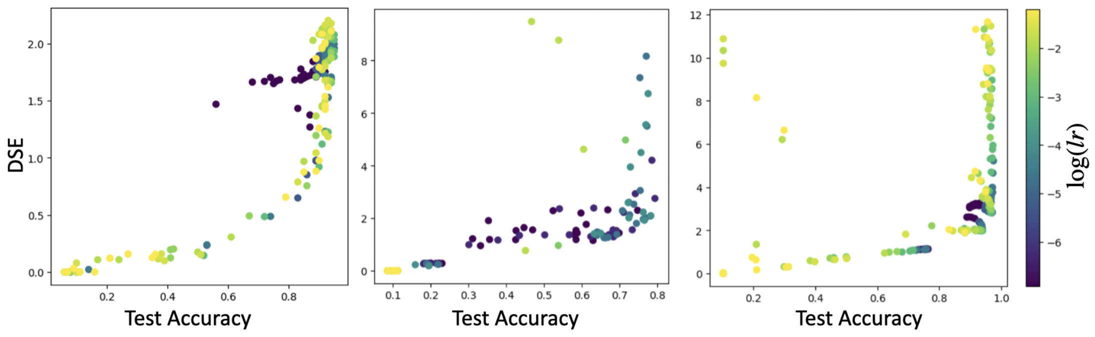


Figure 1: Diffusion Spectral Entropy increases with test accuracy across the three architectures (MLP, ResNet, CNN). Points are colored by learning rate.

2.3.1 Applications of DSE

Evolution of DSE during training In [8], vision networks (including convolutional neural networks and vision transformers) are trained and penultimate layer embeddings are extracted at each epoch. They find that DSE increases in "proper" training (i.e. supervised or contrastive learning on correct labels) as the model performance on classification tasks increases.

Guiding network initialization [8] find that initializing ResNets models at a lower $S_D(Z)$ can improve convergence speed and performance. In fact, starting at higher DSE is an undesirable high-entropy state from which the network attempts to get away before returning to the desirable high-entropy state. This is why we might observe a decrease and then increase in DSE for a high initial $S_D(Z)$, whereas the DSE monotonically increases for a lower initial $S_D(Z)$.

Characterizing neural networks Hypothesizing that neural networks can be largely characterized by the data representations they create, [paper] proposed to generate a low-dimensional manifold of neural networks organized by their hidden-layer representations of a dataset. DSE is computed on the generated penultimate embeddings. High-performance networks across architectures tend to have higher Diffusion Spectral Entropy (DSE) associated with their internal representations, using $t = 0$ because of the low noise in the dataset. 1 indicates this positive relationship between network performance and spread across diffusion matrix dimensions.

The results indicate that high-performing networks exhibit more pronounced cluster structures in their hidden layer embeddings. This pattern is in agreement with the findings of [8]: for k separated clusters, they prove that as t grows to infinity, $S_D(P_z, t)$ tends to $\log(k)$. This makes sense because for graphs that have disconnected components, we would expect to have multiple eigenvectors with non-trivial eigenvalues, or a more even spread of these eigenvalues across the dimensions represented by the eigenvectors. Therefore, we can interpret DSE as measuring the number of significant eigendirections in the data. On the other hand, if the dataset only forms one uniform cluster, since it can be represented by a fully connected graph (i.e. an ergodic process), we would expect the DSE to approach 0 as t tends to infinity since we would have only one non-trivial eigendirection. In other words, the diffusion matrix that represents such data would approach a rank 1 steady-state matrix.

2.4 Graph Neural Networks

Graphs are a fundamental data structure that appear in various domains such as social networks, molecular biology, and recommendation systems. Unlike traditional data structures like grids

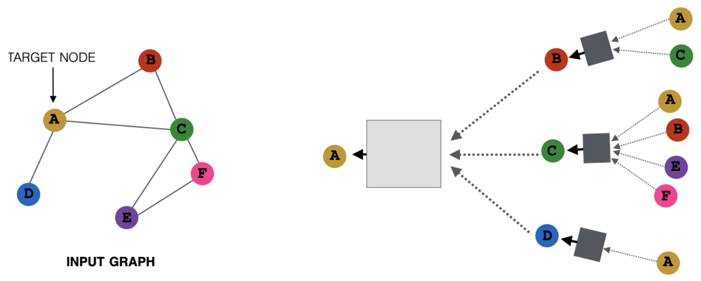


Figure 2: Left: Graph with target node A. Right: We draw the computation graph for A. The first step consists in collecting messages from A's neighbors before aggregating them. Then, the feature vector of node A is updated taken into account the aggregated messages of its neighbors.

(images) or sequences (texts), graphs are inherently irregular, making it difficult to apply conventional deep learning methods directly. Graph Neural Networks (GNNs) have emerged as a powerful class of models designed to operate on graph-structured data by leveraging the relationships between nodes and edges. The key idea behind GNNs is to learn node or graph representations by propagating information through the graph structure in an efficient and scalable manner. The output of a GNN should be a node- or subgraph- or graph-level embedding (depending on the application) on which we can perform further downstream tasks.

GNN Layer The core mechanism of GNNs is the *message-passing* process, which allows nodes to communicate with their neighbors and aggregate information. In this paradigm, each node in the graph updates its state by receiving messages from its neighbors. At layer l, for each node v computes a message:

$$\mathbf{m}_u = \mathrm{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right), u \in \mathcal{N}(v) \cup v \tag{5}$$

where $\mathbf{h}_u$ is the embedding of node u.

Aggregation involves combining the information from neighboring nodes to generate a new embedding for each node (Figure 2).

$$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l)}\left(\mathbf{m}_u^{(l)}, u \in \mathcal{N}(v)\right) \tag{6}$$

Common aggregation functions AGG include sum, mean, and maximum. Once the information has been aggregated, a transformation step typically follows, where the aggregated information is passed through a neural network (often a multi-layer perceptron, or MLP) to update the node's representation. This process is repeated over several iterations (or layers) of message passing, allowing each node to gather information from progressively farther neighbors.

GCN layer The Graph Convolutional Network (GCN) is one of the most well-known GNN architectures. A GCN layer operates by first normalizing the adjacency matrix of the graph and then performing a graph convolution operation to aggregate neighbor information. Formally, for a given node v, the GCN layer updates its feature vector $\mathbf{h}_v^{(l)}$ as:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\deg(v)}\sqrt{\deg(u)}} \mathbf{W}\mathbf{h}_u^{(l)}\right) \tag{7}$$

Here, $\mathbf{h}_v^{(l)}$ denotes the feature vector of node v at layer l, $\mathcal{N}(v)$ is the set of neighbors of v, $\deg(v)$ is the degree of node v, and $\mathbf{W}$ is a learnable weight matrix. The function σ is a non-linear activation function, such as ReLU. The GCN layer effectively aggregates the features from neighboring nodes, normalizing by the degree of each node to prevent over-smoothing caused by large-degree nodes.

GraphSAGE layer While GCN operates on the full graph, GraphSAGE (Graph Sample and Aggregation) offers a more scalable approach, especially for large graphs. Instead of aggregating over all neighbors, GraphSAGE samples a fixed number of neighbors for each node and aggregates their features. The update rule for a GraphSAGE layer is given by:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W} \cdot \text{AGGREGATE}\left(\{\mathbf{h}_u^{(l)} : u \in \mathcal{N}(v)\}\right) \oplus \mathbf{h}_v^{(l)}\right) \tag{8}$$

Where AGGREGATE is an aggregation function, such as mean, LSTM-based aggregation, or pooling. The operation $\oplus$ represents the concatenation of the node's own features with the aggregated features of its neighbors. By introducing the concept of sampling, GraphSAGE alleviates the computational bottleneck of GCN when dealing with very large graphs.

GIN layer Graph Isomorphism Networks (GINs) are a class of GNNs designed to be the most expressive GNN layer in terms of their ability to distinguish between non-isomorphic graphs. This expressiveness comes from using a neural network to model the hash function in the Weisfeiler-Lehman (WL) graph isomorphism test. The WL test is a well-known graph isomorphism test that iteratively refines node labels based on the structure of the graph. GIN approximates this test by employing a learnable neural network to refine node representations during message passing. This makes GIN highly effective for tasks like graph classification, where distinguishing subtle graph structures is crucial.

The core of the GIN message-passing mechanism can be expressed as follows:

$$h_v^{(k+1)} = \text{MLP}\left(\sum_{u \in \mathcal{N}(v)} h_u^{(k)}\right)$$

Here, $h_v^{(k)}$ represents the feature vector of node v at the k-th layer, $\mathcal{N}(v)$ is the set of neighbors of node v, and MLP$(\cdot)$ is a multi-layer perceptron (MLP) that updates the node's feature vector. This process is repeated for multiple layers, where the aggregation function is typically the sum of the neighbor features. This sum-based aggregation method has been shown to be a powerful way to combine node features in graph-based learning tasks.

By using the sum aggregation combined with a powerful MLP update function, GIN is able to capture rich graph structures and distinguish between graph structures that may be indistinguishable for simpler GNNs. As a result, the GIN layer achieves a level of expressiveness equivalent to the WL graph isomorphism test, making it one of the most powerful GNN architectures in practice.

Stacking GNN layers In practice, GNNs are typically composed of multiple layers stacked on top of each other. While stacking multiple GNN layers allows for the aggregation of information over larger neighborhoods, the *receptive field* of each node (i.e., the size of the neighborhood from which it can aggregate information) grows exponentially with the number of layers. As the receptive field increases, node representations can become overly smoothed, losing their distinctiveness. This phenomenon, known as *oversmoothing*, occurs when node features converge to a common value across the entire graph, making it impossible to distinguish between different nodes. The oversmoothing problem is particularly problematic in deep GNNs, where the aggregation process dilutes the node features over several layers.

Skip connections One effective solution to the oversmoothing problem is the use of *skip connections*, which allow the original features of the nodes to bypass the message passing layers and be combined with the transformed features. Skip connections create a mixture of models, where the network can learn to combine the benefits of both deep and shallow GNNs. Specifically, skip connections allow the model to avoid the degenerate behavior of having all node features become the same, while still enabling the network to learn deep representations that capture rich neighborhood information. Mathematically, for a node v, the output of a stacked GNN layer with skip connections can be written as:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W} \cdot \text{AGGREGATE}\left(\{\mathbf{h}_u^{(l)} : u \in \mathcal{N}(v)\}\right) \oplus \mathbf{h}_v^{(l)}\right) \tag{9}$$

In this formulation, the skip connection helps preserve the original features of the node, allowing for both deep and shallow aggregations, depending on the task and depth of the model.

Improving expressiveness with MLP layers To combat oversmoothing and increase the expressiveness of GNNs, one approach is to incorporate MLP layers in the network. MLPs can be added either as pre-processing or post-processing layers to transform node features before or after the aggregation step. By introducing non-linearities and additional transformation layers, MLPs can enhance the capacity of the GNN to model complex relationships, making the aggregation and transformation process more expressive.

2.5 Geometric Scattering

However, GNNs designed for node-level tasks typically perform limited processing from a signal perspective. A key principle in node-level analysis is *homophily* [17], which suggests that nodes tend to be similar to their neighbors. As a result, the goal is to create a hidden representation for each node that changes slowly across neighboring nodes. However, in the context of signal-level tasks, the homophily assumption may not hold. It becomes essential to capture (1) both low-frequency and high-frequency components of the input signal, and (2) both local and global patterns of behavior.

Geometric scattering is a multi-order, multi-scale transform that alternates wavelet transforms and non-linear modulus activations in the form of a deep (typically fixed) network. We can think of scattering as a handcrafted GNN layer, as we design the feature filters, which we refer to as scattering transforms, instead of training them. The presentation of geometric scattering that follows is inspired by [5] and [13].

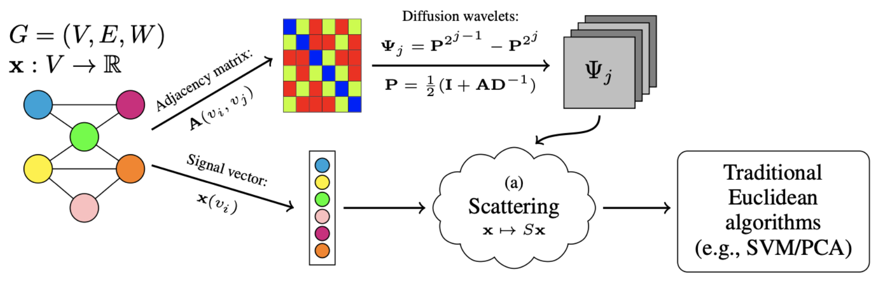
The **lazy random walk operator** is defined as

$$\tilde{\mathbf{P}} = \frac{1}{2}\left(\mathbf{I} + \mathbf{W}\mathbf{D}^{-1}\right).$$

Its eigenvectors and eigenvalues are related to the eigenspectrum of the normalized Laplacian $\mathcal{L}$ by

$$\rho_j = \mathbf{D}^{1/2}\psi_j \quad \text{and} \quad \gamma_j = 1 - \frac{\lambda_j}{2}.$$

The operator $\tilde{\mathbf{P}}$ describes a low-pass operator which, once applied to a signal, retains its low frequencies. In fact, $\tilde{\mathbf{P}}\mathbf{x}$ is the weighted average of $\mathbf{x}(v_l)$ and the values of its neighbors $\mathbf{x}(v_m)$,



$G = (V, E, W)$
$\mathbf{x} : V \to \mathbb{R}$
Adjacency matrix:
$\mathbf{A}(v_i, v_j)$
Signal vector:
$\mathbf{x}(v_i)$
Diffusion wavelets:
$\Psi_j = \mathbf{P}^{2^{j-1}} - \mathbf{P}^{2^j}$
$\mathbf{P} = \frac{1}{2}(\mathbf{I} + \mathbf{A}\mathbf{D}^{-1})$
Ψ_j
(a)
Scattering
$\mathbf{x} \mapsto S\mathbf{x}$
Traditional Euclidean algorithms (e.g., SVM/PCA)

Figure 3: Architecture for using geometric scattering from graph G and signal $\mathbf{x}$ in graph data analysis. Adapted from [13].

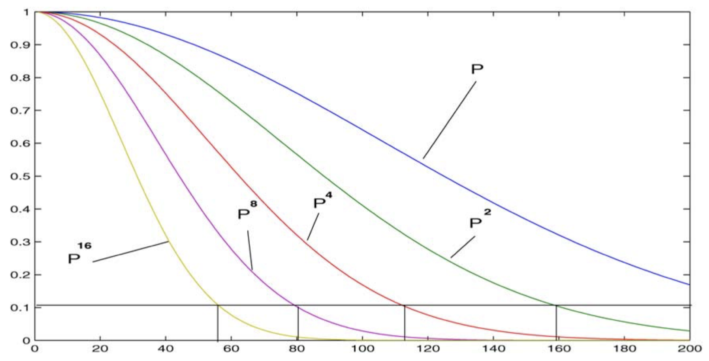


Figure 4: The numerical rank and number of significant eigenvalues of P^t decay at t increases. Adapted from [3].

where $m \in N(v_l)$, for all signals $\mathbf{x} \in \mathbb{R}$. Similarly, $\tilde{\mathbf{P}}^t\mathbf{x}$ is the weighted average of $\mathbf{x}(v_l)$ and the values of $\mathbf{x}(v_m)$, where v_m is within t time steps of v_l. In particular, $\tilde{\mathbf{P}}^t$ preserves zero frequencies while suppressing high frequencies (Figure 4).

Theorem 2.4.

$$\mathbf{y}_{\mathbf{x}_t} = \widehat{\mathbf{y}_\mathbf{x}}(0)\psi_0 + \sum_{k=1}^{n-1} \gamma_k^t \widehat{\mathbf{y}_\mathbf{x}}(k)\psi_k \tag{10}$$

where $\mathbf{y}_\mathbf{x} = D^{-1/2}\mathbf{x}$ *and* $\widehat{\mathbf{x}}(k) = \mathbf{x} \cdot \psi_k$ *is the Fourier transform of* $\mathbf{x}$.

Proof.

$$\mathbf{y}_{\mathbf{x}_t} = \mathbf{D}^{-1/2}\mathbf{x}_t = \mathbf{D}^{-1/2}\tilde{\mathbf{P}}^t\mathbf{x} = \mathbf{D}^{-1/2}\mathbf{D}^{1/2}\sum_{k=0}^{n-1}\widehat{\mathbf{y}_\mathbf{x}}(k)\gamma_k^t\psi_k = \widehat{\mathbf{y}_\mathbf{x}}(0)\psi_0 + \sum_{k=1}^{n-1}\widehat{\mathbf{y}_\mathbf{x}}(k)\gamma_k^t\psi_k$$

□

In order to recover high-frequency values of $\mathbf{x}$, we can utilize multiscale wavelet transforms, which are stable operators in Euclidean space ([9] Lemma 2.12). We can think of the wavelet transform as the difference between two different dyadic scales (powers) of the diffusion operator $\tilde{\mathbf{P}}$. The wavelets act as band-pass filters that capture information at different frequencies and scales. The scattering filters are defined as

$$\mathbf{\Psi_0} = I_n - \tilde{\mathbf{P}} \tag{11}$$

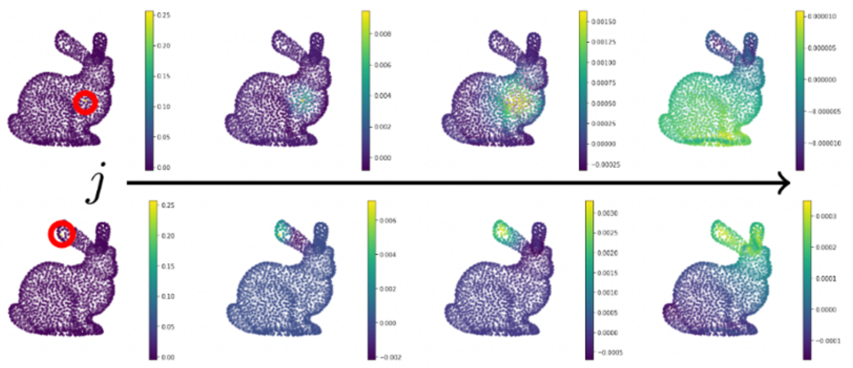


Figure 5: Wavelets $\mathbf{\Psi}_j$ for increasing scale 2^j left to right, applied to Dirac signals centered at two different locations marked by red circles. Vertex colors indicate wavelet values, ranging from yellow (positive values) to blue (negative values). Adapted from [13].

$$\mathbf{\Psi}_j = \tilde{\mathbf{P}}^{2^{j-1}} - \tilde{\mathbf{P}}^{2^{j-1}} \tag{12}$$

$$\mathbf{\Phi}_j = \tilde{\mathbf{P}}^{2^J} \tag{13}$$

where $\mathbf{\Psi}_j$ is referred to as the wavelet matrix in [3].

As such, $\mathbf{\Psi}_j\mathbf{x}(v_l)$ aggregates the signal information from the vertices v_m that are within 2^j steps of v_l but does not average the information, in contrast with the operator $\tilde{\mathbf{P}}^{2^j}$. $\mathbf{\Psi}_j\mathbf{x}$ responds to sharp transitions of $\mathbf{x}$ within a neighborhood of v_l of radius 2^j. The smaller the wavelet scale 2^j, the higher frequencies $\mathbf{\Psi}_j\mathbf{x}$ is able to recover in $\mathbf{x}$. Figure 5 provides some intuition for wavelets.

Theorem 2.5. *The wavelet transform is injective and stable with respect to additive noise.*

Proof. See Theorem 1 in [13]. □

The diffusion scattering transform yields features that are stable to graph structure deformations. Figure 3 summarizes the described architecture for use in data analysis. Geometric scattering is multiscale and therefore incorporates macroscopic graph patterns in every layer.

We can now define a full wavelet filter bank:

$$\mathcal{W}_J := \{\mathbf{\Psi_j}, \mathbf{\Phi}_J\}_{0 \leq j \leq J} \tag{14}$$

The node-level scattering features parameterized by the scattering path $p := (j_1, ..., j_m)$ are:

$$\mathbf{U}_p\mathbf{x} := \mathbf{\Psi}_{j_m}|\mathbf{\Psi}_{j_{m-1}}...|\mathbf{\Psi}_{j_2}|\mathbf{\Psi}_{j_2}\mathbf{x}||...| \tag{15}$$

For whole graph classification tasks, we need graph-level representation. This can be obtained by aggregating node-level features via statistical moments over the nodes of the graph:

$$\mathbf{S}_{p,q}\mathbf{x} = \sum_{i=1}^{n} |\mathbf{U}_p\mathbf{x}(v_i)|^q \tag{16}$$

Geometric scattering embeds graphs into the Euclidean space formed by scattering features and has some nice properties.

Theorem 2.6. $\mathbf{U}_p$ *is equivariant to permutations of the nodes, and* $\mathbf{S}_{p,q}$ *is permutation-invariant.*

Proof. Let Π be a permutation and let $G = \Pi(G)$ be the graph obtained by permuting the vertices of G. $\Pi\mathbf{x}$ is the signal on G'. Let $\mathbf{U}'_p$ is the operation $\mathbf{U}_p$ on G' and $\mathbf{S}'_{p,q}$ is the operation $\mathbf{S}_{p,q}$ on G'.
We want to show that $\mathbf{U}'_p\Pi\mathbf{x} = \Pi\mathbf{U}_p\mathbf{x}$. We have that, for all $0 < j < J$:

$$\mathbf{\Psi}'_j\mathbf{x} = (\Pi\tilde{\mathbf{P}}^{t_j}\Pi^\top - \Pi\tilde{\mathbf{P}}^{t_{j+1}}\Pi^\top)\Pi\mathbf{x} = \Pi\mathbf{\Psi}_j\mathbf{x}$$

We can show this holds for $j = 0$ and $j = J$:

$$\mathbf{\Psi}'_0\Pi\mathbf{x} = (\Pi I_n\Pi^\top - \Pi\tilde{\mathbf{P}}\Pi^\top)\Pi\mathbf{x} = \Pi\mathbf{\Psi}_0\mathbf{x}$$

$$\mathbf{\Phi}'_J\Pi\mathbf{x} = (\Pi\tilde{\mathbf{P}}^{t_J}\Pi^\top)\Pi\mathbf{x} = \Pi\mathbf{\Phi}_j\mathbf{x}$$

Note that $|\Pi\mathbf{x}| = \Pi|\mathbf{x}|$. Then,

$$\mathbf{U}'_p\Pi\mathbf{x} = \mathbf{\Psi}'_{j_m}|\mathbf{\Psi}'_{j_{m-1}}...|\mathbf{\Psi}'_{j_2}|\mathbf{\Psi}'_{j_1}\mathbf{x}||...| = \mathbf{\Psi}'_{j_m}|\mathbf{\Psi}'_{j_{m-1}}...|\mathbf{\Psi}'_{j_2}\Pi|\mathbf{\Psi}_{j_1}\mathbf{x}||...|$$

This inductively gives

$$\mathbf{U}'_p\Pi\mathbf{x} = \Pi\mathbf{U}'_p\mathbf{x}$$

We now show that $\mathbf{S}'_{p,q}\Pi\mathbf{x} = \Pi\mathbf{S}_{p,q}\mathbf{x}$. Note that, for any $q > 0$, $|\Pi\mathbf{x}|^q = \Pi|\mathbf{x}|^q$ and $\sum_j(\Pi\mathbf{x})_j = \sum_j \mathbf{x}_j$ since sums are commutative. Then,

$$\mathbf{S}'_{p,q}\Pi\mathbf{x} = \sum_{i=1}^{n}|\mathbf{U}'_p\Pi\mathbf{x}(v_i)|^q = \sum_{i=1}^{n}|\Pi\mathbf{U}'_p\mathbf{x}(v_i)|^q = \mathbf{S}_{p,q}\mathbf{x}$$

□

2.5.1 Applications of geometric scattering

GRASSY GNNs have become essential tools in drug design and discovery, where molecules are represented as graphs with nodes encoding atomic information and edges representing bonds. Despite their success in predicting molecular properties and binding affinities, generating new molecules with optimized properties remains challenging, particularly due to the limitations of traditional message-passing mechanisms and slow, error-prone reinforcement learning-based graph generation frameworks. The GRASSY (GRAph Scattering SYnthesis network) [2] framework was designed to address these challenges. GRASSY utilizes the geometric scattering transform to capture rich, multi-scale structural information of molecular graphs. This approach creates a permutation-invariant, globally-contextualized representation of molecules, which is then mapped to a structured latent space using a regularized autoencoder. The latent space is further leveraged by a generative adversarial network (GAN) to directly produce molecular graphs with desired properties, bypassing the inefficiencies of step-by-step generation. GRASSY offers a novel, efficient solution for goal-directed drug synthesis by generating meaningful, high-quality molecular structures in a fully structured manner, without the need for sequential or reinforcement learning techniques.

ProtScape Understanding protein dynamics is crucial for studying their biological functions, yet modeling protein motions on timescales of microseconds to milliseconds remains challenging. To address this, the ProtSCAPE (Protein Transformer with Scattering, Attention, and Positional Embedding) [14] framework integrates geometric scattering with transformer-based attention mechanisms to capture protein dynamics from molecular dynamics (MD) simulations. By

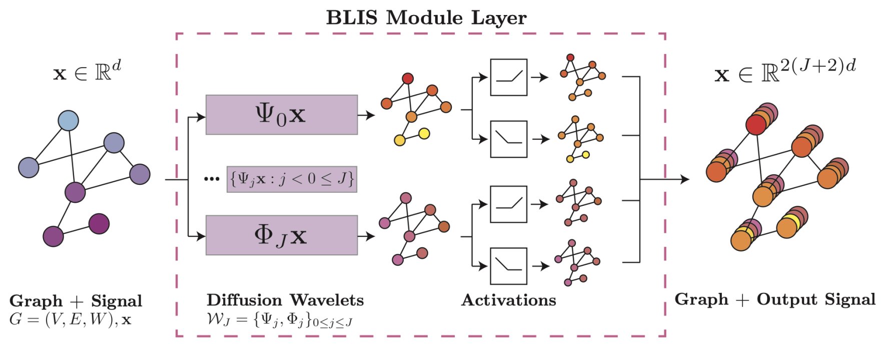

BLIS Module Layer
$\mathbf{x} \in \mathbb{R}^d$
$\Psi_0\mathbf{x}$
$\{\Psi_j\mathbf{x} : j < 0 \leq J\}$
$\Phi_J\mathbf{x}$
$\mathbf{x} \in \mathbb{R}^{2(J+2)d}$
Graph + Signal
$G = (V, E, W), \mathbf{x}$
Diffusion Wavelets
$\mathcal{W}_J = \{\Psi_j, \Phi_j\}_{0 \leq j \leq J}$
Activations
Graph + Output Signal

Figure 6: The BLIS module: it applies multiscale diffusion wavelet transform to the input signal $\mathbf{x}$ and two activation functions. The output is a multivariate signal with $2(J + 2)$ times the original signal dimension. Adapted from [16].

conceptualizing proteins as graphs, ProtSCAPE applies multi-scale geometric scattering to extract structural features and combines them with dual attention over residues and amino acids, enabling the model to generate latent representations of protein trajectories. ProtSCAPE also incorporates a regression network to enforce temporal coherence in the learned representations, improving its ability to generalize across both short and long MD trajectories, as well as wild-type and mutant proteins. This novel approach surpasses traditional methods by offering more precise and interpretable upsampling of protein dynamics, identifying flexible residues critical for conformational transitions, and providing insights into phase transitions and stochastic switching. ProtSCAPE's ability to visualize MD trajectories in a low-dimensional latent space and predict key protein properties makes it a powerful tool for understanding protein conformational changes and designing novel protein mutants.

2.5.2 BLISNet

Note that the proposed geometric scattering is not injective because of the absolute value operation. To remedy this downfall, [16] proposes bi-Lipschitz scattering (BLIS), which uses `ReLU` and reflected `ReLU` instead of the modulus operation to preserve injectivity:

$$\mathbf{U}_p\mathbf{x} = M\mathbf{\Psi}_{j_m}...M\mathbf{\Psi}_{j_1}\mathbf{x} \tag{17}$$

where $M\mathbf{x} = \texttt{ReLU}(\mathbf{x}) + \texttt{ReLU}(-\mathbf{x})$. The BLIS module inherits geometric scattering's nice properties, including permutation equivariance and conservation of energy. The following result is proved in the paper (Theorem 3.2 and Corollary 3.1). The BLIS module is defined as:

$$B[j_1, k_1, ..., j_m, k_m](\mathbf{x}) = \sigma_{k_m}(F_{j_m}\sigma_{k_{m-1}}(F_{j_{m-1}} \cdots \sigma_{k_1}(F_{j_1}\mathbf{x})) \cdots) \tag{18}$$

where we write the wavelet frame $\mathcal{W}_J$ as $\mathcal{F} = \{F_j\}_{j=0}^{J+1}$ where $F_j = \Psi_j$ for $0 \leq j \leq J$ and $F_{J+1} = \Phi_J$, and $m \geq 1$ is the depth of the network. Let $\mathbf{B}_m(\mathbf{x})$ be the set of all $B[j_1, k_1, ..., j_m, k_m]$.

Theorem 2.7. *B_m is injective on $\mathbb{R}^n$.*

This property renders the BLIS module more expressive than classical geometric scattering. BLIS is also able to preserve the energy (i.e. norm) of the input signal in every layer, whereas previous approaches to scattering only showed energy conservation when summing over all

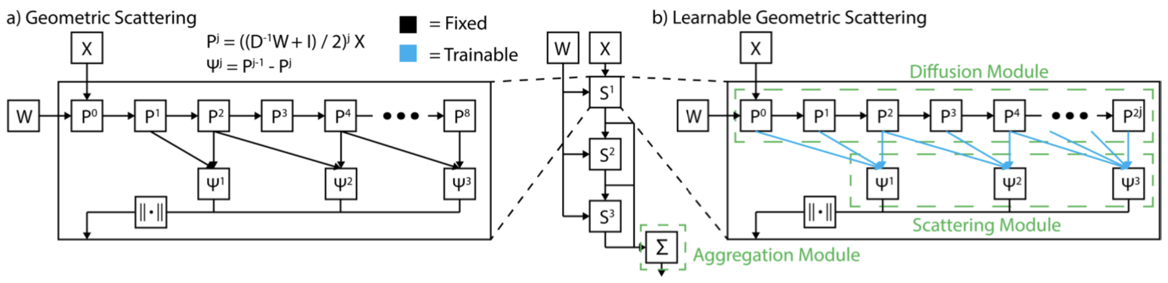


Figure 7: The LEGS module learns to select the appropriate scattering scales from the data. Adapted from [13].

layers (Theorem 3.4 in [16]). The BLIS-Net architecture integrates a BLIS module layer with a moment-aggregation module which aggregates BLIS features across nodes, before performing dimensionality reduction to generate an embedding on which to perform graph classification. [16] shows that BLIS-Net achieves higher performance on synthetic and real-world data sets derived from traffic and fMRI data.

2.5.3 LEGS

[13] proposes to use geometric scattering as a trainable GNN module, where the scales, or powers of the diffusion operator in the wavelet bank, are learnable. This LEarnable Geometric Scattering (LEGS) method allows for the adaptive tuning of wavelets and thus the learning of longer-range graph relations. In contrast, the deterministic geometric scattering methods described above simply use dyadic scales. The architecture of the LEGS module is summarized in Figure 7.

The diffusion submodule The diffusion matrix $\tilde{P}$ is iteratively multiplied to give

$$[\tilde{\mathbf{P}}\mathbf{x}, \tilde{\mathbf{P}}^2\mathbf{x}, ..., \tilde{\mathbf{P}}^m\mathbf{x}]$$

. In LEGS, these operations are implemented using m recurrent neural network (RNN) modules.

The scattering submodule This module selects $J \leq m$ diffusion scales to construct the wavelet filter bank $\mathcal{W}_J$. The scale selection step uses a selection matrix $\mathbf{F} \in \mathbb{R}^{J \times m}$, where each row $\mathbf{F}_{(j,\cdot)}, j \in [J]$ is a one-hot encoding that identifies the diffusion scale of $\tilde{\mathbf{P}}^{t_j}$. Let $\theta_j \in \mathbb{R}^m$ be the rows of a trainable weight matrix $\mathbf{\Theta}$:

$$\mathbf{F} := [\text{softmax}(\theta_1), ..., \text{softmax}(\theta_J)]^\top \tag{19}$$

Note that the softmax function is meant to approximate a one-hot encoding. The rows of $\mathbf{F}$ are further ordered according to the position of the leading one activated in every row. The filter bank $\tilde{\mathcal{W}}_\mathbf{F} := \{\tilde{\mathbf{\Psi}}_j, \tilde{\mathbf{\Phi}}_J\}_{j=0}^{J-1}$ is constructed with the filters

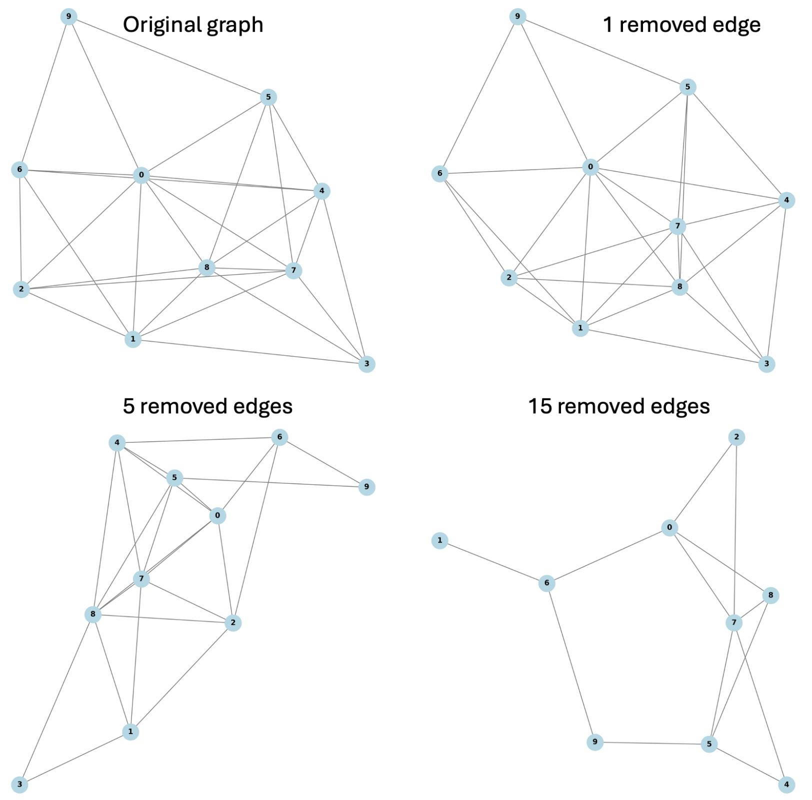


Figure 8: Generated 10-node graph. We perturb an increasing number of edges to create different graphs from the original.

$$
\begin{aligned}
\tilde{\mathbf{\Psi}}_0\mathbf{x} &= \mathbf{x} - \sum_{t=1}^{m} \mathbf{F}_{(0,1)}\tilde{\mathbf{P}}^t\mathbf{x} \\
\tilde{\mathbf{\Psi}}_j\mathbf{x} &= \sum_{t=1}^{m} \left[\mathbf{F}_{(j,t)}\tilde{\mathbf{P}}^t\mathbf{x} - \mathbf{F}_{(j+1,t)}\tilde{\mathbf{P}}^t\mathbf{x}\right],\ 1 \leq j \leq J-1 \\
\tilde{\mathbf{\Phi}}_J\mathbf{x} &= \sum_{t=1}^{m} \mathbf{F}_{(J,t)}\tilde{\mathbf{P}}^t\mathbf{x}
\end{aligned}
\tag{20}
$$

The aggregation submodule The node-level features are aggregated into graph-level features using statistical moment aggregation, as shown in equation 16.

3 Methods and Experiments

Graph dataset generation We generate random graphs (binomial graphs with $p = 0.5$) with 10, 30, and 50 nodes. Each graph is randomly perturbed with an increasing number of edges, allowing us to create datasets with different variances. This results in a set of progressively more varied datasets. Additionally, the graphs are equipped with Dirac signals at each node. We generate a total of 180 datasets, each with 1,000 graphs. We visualize an example in Figure 8.

Scattering We apply two versions of scattering: the ”vanilla” scattering method that iteratively applied wavelets and takes the absolute value; and the injective version of scattering, as proposed for the BLIS module. To get a representation of each graph, instead of aggregating the node representations as in 16, we concatenate the node-level representations obtained in 15. Before computing DSE on the obtained embeddings, we perform dimensionality reduction using an autoencoder.

GNN autoencoder For comparison, we train simple graph autoencoders (GAE). The encoder is implemented as a two-layer GCN and the decoder is an inner-product decoder as proposed in [7]. We compute DSE on the output of the trained encoder.

Molecular dataset We apply our method to a molecular dataset, ZINC [6]. ZINC contains drug-like molecules organized into tranches by molecular weight (abbreviated mol. wt.), solubility (logP value), reactivity and commercial availability.

Hyperparameters and experiment set-up

- Each model is trained for 100 epochs.
- We use the Adam optimizer and set the learning rate to 0.01.
- We use cross-entropy loss for binary and multi-class classification.
- Both autoencoders are trained on the entire datasets and have a latent dimension of 32.
- The autoencoder used on scattering coefficients is trained with a mean-squared error (MSE) reconstruction loss. On the other hand, the graph autoencoder is trained using a binary cross-entropy reconstruction loss.

4 Results

Our analysis involved computing Diffusion Spectral Entropy (DSE) across several synthetic datasets consisting of graphs with 10, 30, and 50 nodes. Each dataset contained graphs that were progressively perturbed by increasing numbers of edges. The results of DSE calculations are presented in Tables 1, 2, and 3. As shown in Figures 9 and 10, DSE is able to effectively capture the variance of the graph datasets. As we perturb an increasing number of edges, DSE increases, which reflects the heterophily of the dataset. This is true across multiple graph sizes. However, Figure 11 shows that the embeddings generated by the GAE encoder are not as reliable: the correlation between dataset variability and DSE is not clear. In particular, the first and third plot show that some datasets that are less diverse (i.e. less edges have been altered) have higher DSE than more diverse datasets. When computed on GNN embeddings, DSE does not effectively capture the variance of the dataset.

The results suggest that geometric-scattering methods preserve the underlying structure of graph datasets better than traditional GNNs. The relative DSE values computed on geometric scattering coefficients effectively reflect the diversity of graph structures, irrespective of the number of nodes or node features. We visualize the emebddings created by geometric scattering in Figure 12.

We apply our method to the ZINC dataset, which contains drug-like molecules organized into tranches. We choose to focus our analysis on two tranches: `BBAB` and `JBCD`. We compute Diffusion Spectral Entropy (DSE) on the scattering embeddings for each tranche separately, as well as for the combined tranches. The embeddings are visualized in Figure 13 using PHATE,

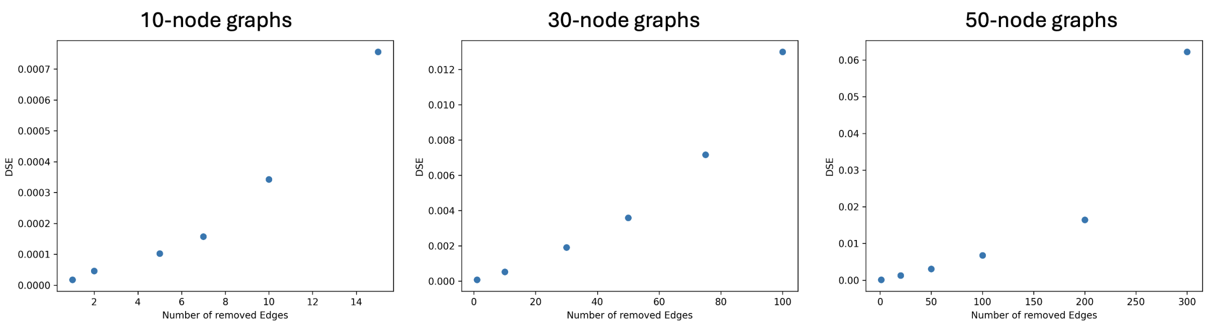


Figure 9: Diffusion spectral entropy computed on the dimensionality-reduced node-level geometric scattering coefficients. Traditional scattering, as shown in Equation 15, is used here.

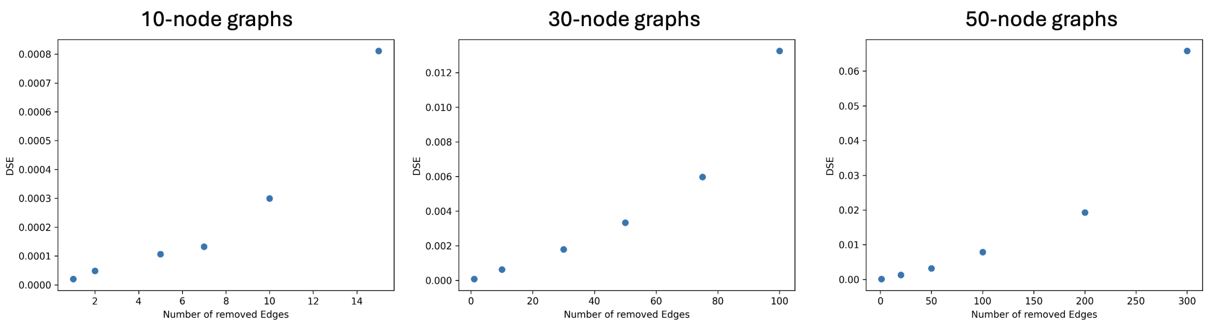


Figure 10: Diffusion spectral entropy computed on the dimensionality-reduced node-level geometric scattering coefficients. Here we compute the scattering features using the injective BLIS method as shown in Equation 16.

which effectively illustrates the clustering of the embeddings by tranche. The DSE values are reported in Table 4. The results show that the embeddings are distinctively clustered based on their respective tranches. In addition, the DSE is higher when considering the mixed tranches, indicating an increased variability and richness in the embedding space when both tranches are combined.

5 Conclusion and Discussion

In this study, we have introduced a novel approach for computing entropy over a set of graphs, leveraging Diffusion Spectral Entropy (DSE) combined with geometric scattering as an effective

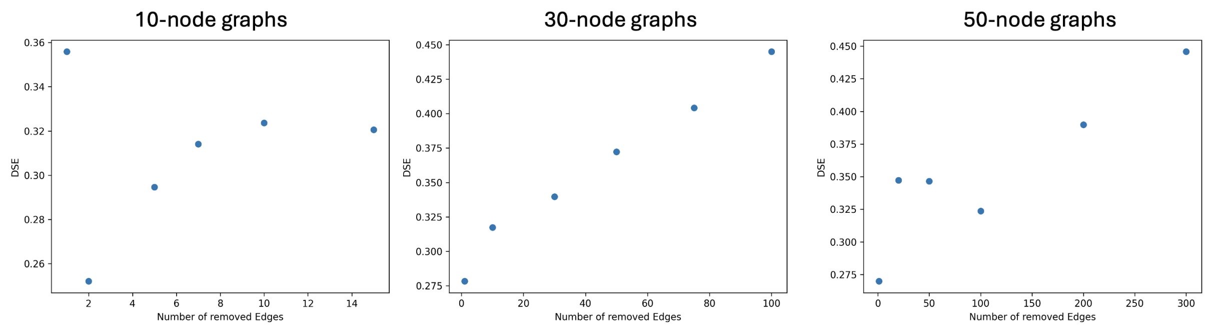


Figure 11: Diffusion spectral entropy computed on embeddings generated by the GAE encoder.

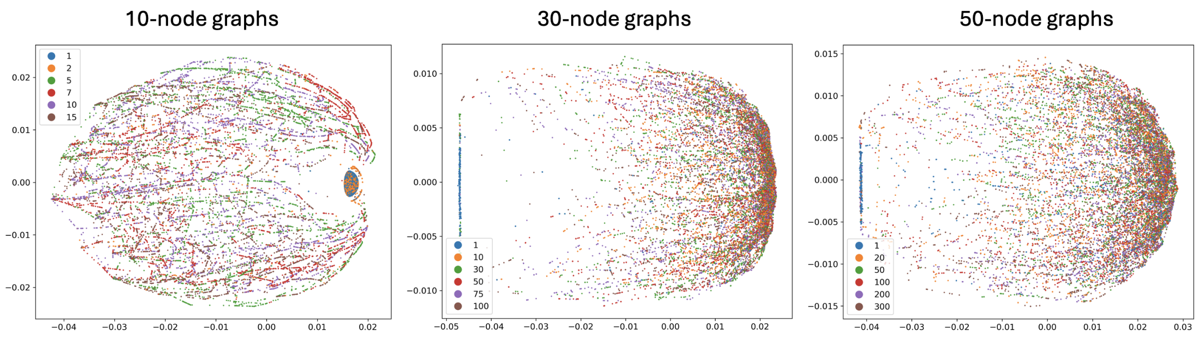


Figure 12: We visualize the dimensionality-reduced scattering coefficients using PHATE [11], a diffusion geometry based dimensionality reduction method. Each point represents a graph. The points are colored by the number that were removed from the original graph.

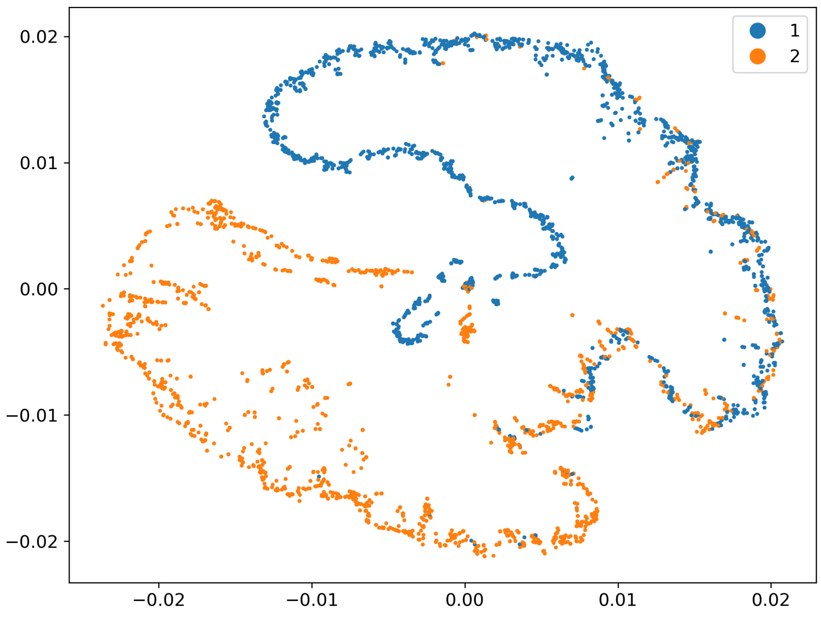


Figure 13: PHATE visualization of the geometric-scattering based embeddings generated by mixing two tranches of the ZINC dataset. Label 1 corresponds to `BBAB` and label 2 corresponds to `JBCD`.

Removed edges	1	2	5	7	10	15
Scattering	(1.73 ± 0.88) e-05	(4.57 ± 1.85) e-05	(1.027 ± 0.44) e-04	(1.57 ± 0.83) e-04	(3.42 ± 2.43) e-04	(7.56 ± 6.33) e-04
BLIS	(1.99 ± 0.92) e-05	(4.83 ± 2.29) e-05	(1.06 ± 0.50) e-04	(1.32 ± 0.74) e-04	(2.99 ± 1.86) e-04	(8.11 ± 6.95) e-04
GAE	(3.56 ± 3.19) e-01	(2.52 ± 0.33) e-01	(2.95 ± 0.26) e-01	(3.14 ± 2.45) e-01	(3.24 ± 0.26) e-01	(3.21 ± 0.40) e-01

Table 1: DSE values of the 10-node graph dataset embeddings. Each DSE value is averaged over 10 datasets generated by altering 10 binomial graphs.

Removed edges	1	10	30	50	75	100
Scattering	(7.29 ± 2.32) e-05	(5.23 ± 1.13) e-04	(1.91 ± 0.58) e-03	(3.59 ± 1.75) e-03	(7.16 ± 3.44) e-03	(1.30 ± 0.62) e-02
BLIS	(7.77 ± 2.32) e-05	(6.24 ± 2.54) e-04	(1.79 ± 0.35) e-03	(3.34 ± 1.43) e-03	(5.97 ± 2.30) e-03	(1.33 ± 0.54) e-02
GAE	(2.78 ± 0.15) e-01	(3.17 ± 0.43) e-01	(3.40 ± 0.35) e-01	(3.72 ± 0.36) e-01	(4.04 ± 0.30) e-01	(4.45 ± 0.27) e-01

Table 2: DSE values of the 30-node graph dataset embeddings. Each DSE value is averaged over 10 datasets generated by altering 10 binomial graphs.

tool for quantifying the variability within graph datasets. Our experiments on both synthetic and real molecular datasets have demonstrated that our method effectively captures the diversity inherent in the datasets by analyzing the graphs' structure and their respective properties.

The results reveal that geometric scattering, compared to traditional Graph Neural Networks (GNNs), better preserves the intrinsic geometric and topological structure of graphs. In particular, geometric scattering successfully embodies both low-frequency and high-frequency information and spans both local and global behaviors within the graph data. This superior performance is evidenced by the relative increased Diffusion Spectral Entropy (DSE) observed in our experiments, which correlates with the increased perturbation and diversity in the graph datasets. These findings establish the potential of geometric scattering in conjunction with DSE as a robust framework for analyzing the variability of graph datasets across various applications.

Furthermore, the application of our method to the molecular dataset from the ZINC database illustrates its practical utility in real-world scenarios. By accurately reflecting the significant variance among molecular structures, our approach proves instrumental in fields like drug design and molecular dynamics, where understanding the structural variability is crucial.

Future work will focus on proving additional properties of geometric scattering, including its effectiveness in capturing the geometric and topological features of graph data, extending the work done in [1].

References

[1] Dhananjay Bhaskar et al. *Learning graph geometry and topology using dynamical systems based message-passing*. 2024. arXiv: `2309.09924 [cs.LG]`. URL: `https://arxiv.org/abs/2309.09924`.

Removed edges	1	20	50	100	200	300
Scattering	(9.58 ± 3.42) e-05	(1.29 ± 0.34) e-03	(3.09 ± 1.10) e-03	(6.79 ± 2.89) e-03	(1.64 ± 0.65) e-02	(6.22 ± 3.90) e-02
BLIS	(1.09 ± 0.35) e-04	(1.31 ± 0.69) e-03	(3.20 ± 1.44) e-03	(7.88 ± 8.70) e-03	(1.93 ± 6.42)e-02	(6.58 ± 6.42) e-02
GAE	(2.70 ± 0.08) e-01	(3.47 ± 0.85) e-01	(3.47 ± 0.24) e-01	(3.24 ± 0.08) e-01	(3.90 ± 0.10) e-01	(4.46 ± 0.12) e-01

Table 3: DSE values of the 50-node graph dataset embeddings. Each DSE value is averaged over 10 datasets generated by altering 10 binomial graphs.

Tranche 1: `BBAB`	Tranche 2: `JCBCD`	Combined Tranches
3.45e-05	4.48e-05	4.30e-05

Table 4: DSE values of two ZINC tranches scattering embeddings.

[2] Dhananjay Bhaskar et al. *Molecular Graph Generation via Geometric Scattering*. 2021. arXiv: 2110.06241 [cs.LG]. URL: https://arxiv.org/abs/2110.06241.

[3] Ronald R. Coifman and Stéphane Lafon. "Diffusion maps". In: *Applied and Computational Harmonic Analysis* 21.1 (2006), pp. 5–30.

[4] Fernando Gama, Alejandro Ribeiro, and Joan Bruna. *Diffusion Scattering Transforms on Graphs*. 2018. arXiv: 1806.08829 [cs.LG]. URL: https://arxiv.org/abs/1806.08829.

[5] Feng Gao, Guy Wolf, and Matthew Hirn. "Geometric scattering for graph data analysis". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2122–2131.

[6] John J. Irwin and Brian K. Shoichet. "ZINC A Free Database of Commercially Available Compounds for Virtual Screening". In: *Journal of Chemical Information and Modeling* 45.1 (2005). PMID: 15667143, pp. 177–182. DOI: 10.1021/ci049714+. eprint: https://doi.org/10.1021/ci049714+. URL: https://doi.org/10.1021/ci049714+.

[7] Thomas N. Kipf and Max Welling. *Variational Graph Auto-Encoders*. 2016. arXiv: 1611.07308 [stat.ML]. URL: https://arxiv.org/abs/1611.07308.

[8] Danqi Liao et al. "Assessing neural network representations during training using noise-resilient diffusion spectral entropy". In: *2024 58th Annual Conference on Information Sciences and Systems (CISS)*. IEEE. 2024, pp. 1–6.

[9] Stéphane Mallat. *Group Invariant Scattering*. 2012. arXiv: 1101.2286 [math.FA]. URL: https://arxiv.org/abs/1101.2286.

[10] Yimeng Min et al. *Can Hybrid Geometric Scattering Networks Help Solve the Maximum Clique Problem?* 2022. arXiv: 2206.01506 [cs.LG]. URL: https://arxiv.org/abs/2206.01506.

[11] Kevin R. Moon et al. "Visualizing structure and transitions in high-dimensional biological data". In: *Nat Biotechnol* 37.12 (2019), pp. 1482–1492.

[12] D. I. Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE Signal Processing Magazine* 30.3 (May 2013), pp. 83–98. ISSN: 1053-5888. DOI: 10.1109/msp.2012.2235192. URL: http://dx.doi.org/10.1109/MSP.2012.2235192.

[13] Alexander Tong et al. *Learnable Filters for Geometric Scattering Modules*. 2022. arXiv: 2208.07458 [cs.LG]. URL: https://arxiv.org/abs/2208.07458.

[14] Siddharth Viswanath et al. *ProtSCAPE: Mapping the landscape of protein conformations in molecular dynamics*. 2024. arXiv: 2410.20317 [cs.LG]. URL: https://arxiv.org/abs/2410.20317.

[15] Zonghan Wu et al. “A Comprehensive Survey on Graph Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (Jan. 2021), pp. 4–24. ISSN: 2162-2388. DOI: 10.1109/tnnls.2020.2978386. URL: http://dx.doi.org/10.1109/TNNLS.2020.2978386.

[16] Charles Xu et al. *BLIS-Net: Classifying and Analyzing Signals on Graphs*. 2023. arXiv: 2310.17579 [cs.LG]. URL: https://arxiv.org/abs/2310.17579.

[17] Jiong Zhu et al. *Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs*. 2020. arXiv: 2006.11468 [cs.LG]. URL: https://arxiv.org/abs/2006.11468.

[18] Dongmian Zou and Gilad Lerman. “Graph convolutional neural networks via scattering”. In: *Applied and Computational Harmonic Analysis* 49.3 (Nov. 2020), pp. 1046–1074. ISSN: 1063-5203. DOI: 10.1016/j.acha.2019.06.003. URL: http://dx.doi.org/10.1016/j.acha.2019.06.003.